Simple Hash Function Using Stack with Page Replacement Principle

Sivakumar.T, Sowmya .P, Utharaa.S

***Abstract*— Hash Function is a special class of function that has certain properties which make it suitable for use in cryptography. Cryptographic hash functions are widely used in digital signatures, message authentication codes (MAC) and also other forms of authentication. They are widely used insecurity protocols, passwords and pseudorandom number generation. MD5, SHA, RIPEMD and Whirlpool are some of the hash functions already available in real time applications. These algorithms involve more number of operations and iterations and thereby consume more time. In this paper, a new hash function using stack data structure with page replacement principle is proposed. From the experimental, it is observed that the proposed method is simple and easy to implement for less volume of data.**

***Index Terms*— Cryptography, Encryption, Stack**

I. INTRODUCTION

Hash functions are extremely useful and appear in almost all information security applications. A cryptographic hash function is a hash function which takes an input of arbitrary size and returns a fixed-size alphanumeric string, which is called the hash code. Practical applications include message integrity checks, digital signatures, authentication, and various information security applications. A hash function takes a string of any length as input and produces a fixed length string which acts as a kind of "signature" for the data provided [1]. A person knowing the "hash value" is unable to know the original message, but only the person who knows the original message can prove the "hash value" is created from that message.

A cryptographic hash function should behave as much as possible like a random function while still being deterministic and efficiently computable[3].A cryptographic hash function is considered "insecure" from a cryptographic point of view, if either of the following is computationally feasible[1]:

a) Finding a (previously unseen) message that matches a given hash values.
b) Finding "collisions", in which two different messages have the same hash value.

An attacker who can find any of the above computations can use them to substitute an authorized message with an unauthorized one. Ideally, it should be impossible to find two different messages whose digests ("hash values") are similar; nor would one want an attacker to be able to learn anything useful about a message given only its digest("hash values")[1].

In the proposed method, the concept of stack along with page replacement principle, used for memory management, is utilized to develop a new hash function. The hash function proposed here is a one way function, so it won't be possible to get the input from the hash value. Page replacement principle is used so that there won't be any repetition of characters in the hash code.

Sivakumar.T, Department of Information Technology PSG College of Technology, Coimbatore-641004, India
Sowmya .P, Department of Information Technology PSG College of Technology, Coimbatore-641004, India
Utharaa.S, Department of Information Technology PSG College of Technology, Coimbatore-641004, India

II. LITERATURE SURVEY

Cryptographic hash functions are an important to achieve certain security goals such as authenticity, digital signatures, digital time stamping, and entity authentication. They are also strongly related to other important cryptographic tools such as block ciphers and pseudorandom functions [5].There are several methods to use a block cipher to build a cryptographic hash function, specifically a one-way compression function. The methods resemble the block cipher modes of operation usually used for encryption. Many well-known hash functions, including MD4, MD5, SHA-1 and SHA-2 are built from block-cipher-like components designed for the purpose, with feedback to ensure that the resulting function is not invertible [2].

A standard block cipher such as AES can be used in place of these custom block ciphers; that might be useful when an embedded system needs to implement both encryption and hashing with minimal code size or hardware area. However, that approach can have costs in efficiency and security [4].

Slightly simplified versions of the hash functions are surprisingly weak: whenever symmetric constants and initialization values are used throughout the computations and modular additions are replaced by exclusive or operations, symmetric messages hash to symmetric digests [7]. Therefore the complexity of collision search on these modified hash functions potentially becomes as low as one wish[6].A hash function h which maps a message of any length to strings of some fixed length is called to be collision free, but such that finding x,y with h(z)=h(y) is a hard problem. The need for such functions to ensure data integrity and for digital signatures is well known [8].

The RIPEMD is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by open research community and generally known as a family of European hash functions. The set includes RIPEMD, RIPEMD-128, and RIPEMD-160. There also exist 256, and 320-bit versions of this algorithm. Original RIPEMD (128 bit) is based upon the design principles used in MD4 and found to provide questionable security [9].

There had been a lot of tweaks and variants in the MD and the SHA series mostly by increasing the length of the message digest. There are approaches to find collisions in MD5 and break other hash functions like RIPEMD, HAVAL, MD4 and SHA-0 by using differential attacks. This has led to the recent development of many other cryptography hash functions,

each having its own strengths and weaknesses, aiming to be the "one" which is secure against birthday attacks, cube testers, differential cryptanalysis and several other attacks [10].

In this paper, a new and simple hash function using the concept of stack and page replacement algorithm is proposed.

III. Proposed Hash Function

In this section, the working model of the proposed hash function is described with a flowchart. In the proposed hash function, the stack data structure is used for permutation, which shuffles the characters in the input data. The character occurrence is also taken into account to avoid collision. The size of stack is maintained as 'n' and the elements are added based on stack operation principle. Value of 'n' is the user choice and this denotes the size of the final hash code generated by the proposed method.

A character is read from the input data, its occurrence is checked in the count array and if it occurs for the first time then it is inserted into the stack. If it has already occurred, then its occurrence value is XORed with all values already present in the stack. This is the principle of page replacement algorithm where we will not replace a page which is already available in frame. The process is repeated for entire message. The final hash code is the value in the entire stack. The working model of the proposed hash function is shown in Fig 1.

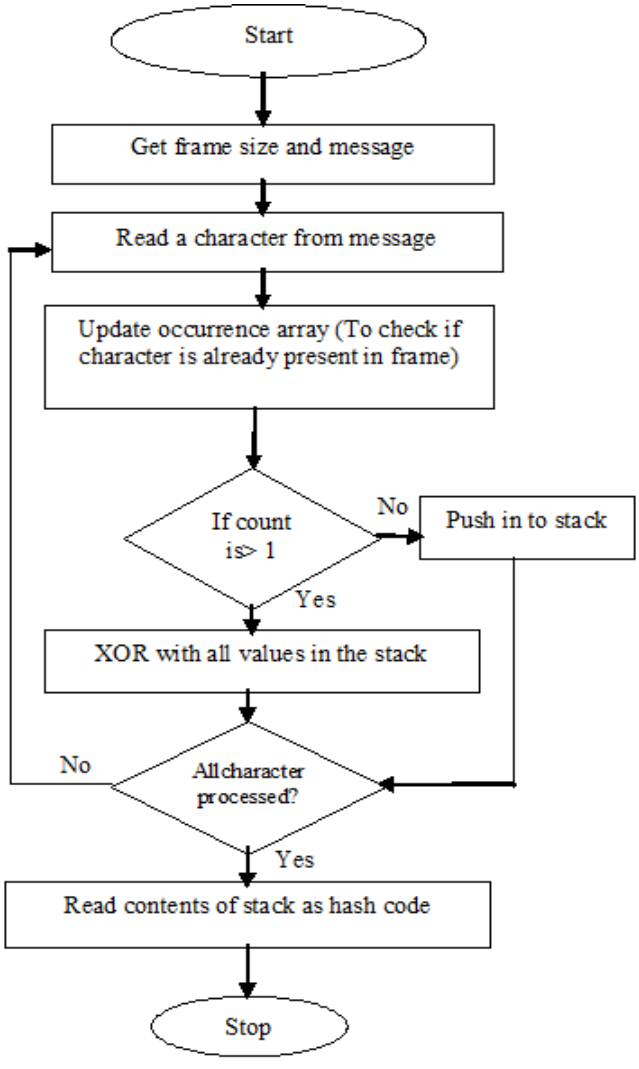

Fig.1. Flow Diagram of Proposed Hash Function

A. *Illustration of Proposed Hash Function*

Let the input message be "meet me after toga party". Compute the occurrence array as shown in Fig 2

a	b	c	d	e	f	g
3	0	0	4	1	1	0

h	i	j	k	l	m	n
0	0	0	0	0	2	0

o	p	q	r	s	t	u
1	1	0	2	0	4	0

v	w	x	y	z
0	0	0	1	0

Fig. 2. Occurrence Array

Input the frame size and start inserting the characters of the message inside the stack. For implementation the ASCII value of the input message is used. Insert 'm' and 'e' first. Then insert 'e'. Since 'e' already exist, XOR its count value with all the elements in the stack. Next insert the character 't'.

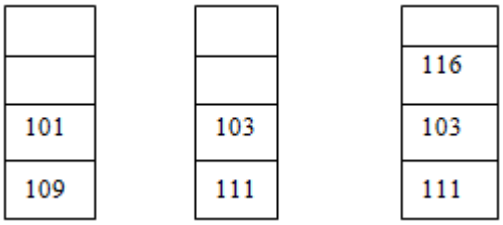

Fig.3.Stack after inserting the characters m,e,e,t

'm' already exists XOR its count value with all the elements of the stack as shown in Fig 4.Next insert the character 'e' and XOR the contents of the stack with its count value. Next, insert the character 'a'.

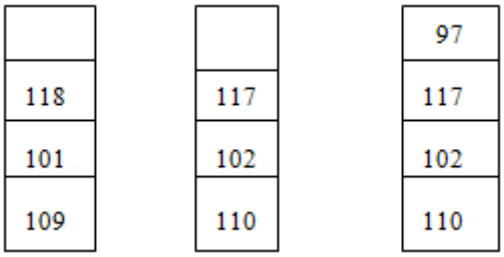

Fig. 4.Stack after inserting the characters m,e,a

Insert the next character 'f'. As 't' already exists XOR its count value with all the elements of the stackas shown in Fig 5.Next insert the character 'e' and XOR the contents of the stack with its count value.

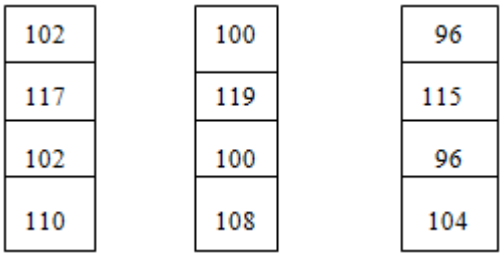

Fig.5.Stack after inserting the characters f,t,e

Insert the next character 'r'. As 't' already exists XOR its count value with all the elements of the stack as shown in Fig 6.Next insert the character 'o'.

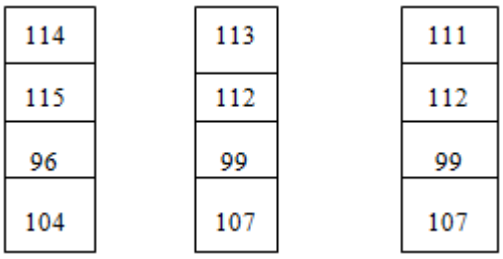

Fig.6.Stack after inserting the characters r,t,o

Insert the next character 'g'. As 'a' already exists XOR its count value with all the elements of the stack as shown in Fig 7.Next, insert the character 'p'.

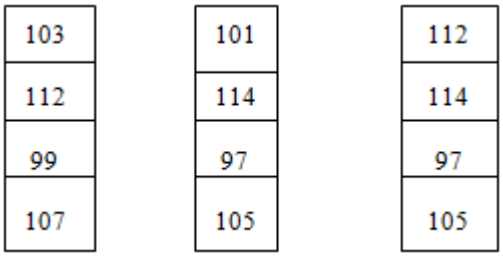

Fig.7.Stack after inserting the characters g,a,p

As 'a' already exists XOR its count value with all the elements of the stack as shown in Fig 8.Next insert the character 'r' and XOR the contents of the stack with its count value. Then insert the character 't'. Next insert the last character 'y'. Finally read the contents of the stack to get the hash code. Here the hash code for the given input message is "lfvc".

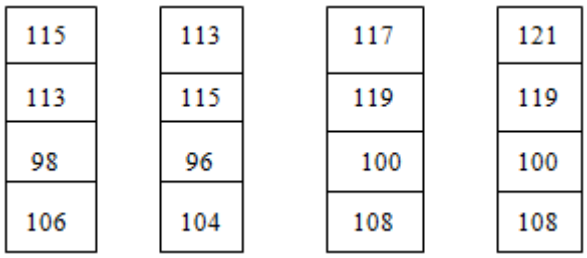

Fig.8.Stack after inserting the characters a,r,t,y

B. Algorithm

Input: message and the hash code size, Output: hash code.

Step 1: Start
Step 2: Input the message to which the hash code is to be computed.
Step 3: Create and initialize the occurrence array.
Step 4: Input the hash code size.
Step 5: Create a stack based on the required hash code size.
Step 6: Read character from input message and use its ASCII value.
Step7: Occurrence of each character is maintained in occurrence array by incrementing count value.
Step 8: Push the character into the stack if its occurrence value is 1. (i.e., it is not in frame).
Step 9: If the count value of the character is > 1, then XOR all the existing characters in the stack with the occurrence value.
Step 10: Repeat the steps 6 to 9 until the entire message is processed.
Step 11: Read the value from the entire stack to get the hash code of the given message.
Step 10: Store the hash code and stop the process.

C. Advantages of Proposed Hash Function

a. This hash function can be used in simple mail applications.

b. It can also be used in our computers to store password.

c. The proposed hash function can also be used in SMS applications.

d. Variable sized hash code can be generated as per user requirement.

IV. Experimental Results

The proposed method is experimented using C++ language and the system configuration is Processor Intel i5-5200U CPU, Clock speed 2.2 GHz, RAM 4GB and the operating system is Windows. The obtained result of proposed method by using various input messages is given in Table 1.

Table – 1Experimental Result

Frame Size	Message	Hash Code	Hash Code (in Hex)
4	Meetmeaftertogaparty	Lfvc	4c667663
8	Meetmeaftertogaparty	Lfvcbuhb	4c667663627 56862
4	Meetmeintogaparty	j'pd	6a927064
4	Indiaisindependent	inf'	696e6692

The obtained hash code varies based with respect to the input message and the number of frames. Currently, the proposed model is tested with character messages forexperimental purpose. This can be extended to numerical and special characters.

V. Conclusion

In this paper, a new hash function is developed using stack and the principle of page replacement. This kind of hash function can be used for hashing small file, SMS and e-mail messages. The proposed hash function gives different hash values for inputs even with the minor difference. It is also possible to change the frame size based upon the required length of the hash code. This work can be further extended to test the properties of the hash function. In future, the proposed method is validated based on the requirements of hash functions.

REFERENCES

[1] William Stallings, "Cryptography and Network Security-Principles and Practice", Pearson Education, New Delhi,2013.

[2] Priyanka Vadhera and Bhumika Lall, "Review Paper on Secure Hashing Algorithm and Its Variants", International Journal of Science and Research, Volume 3 , Issue 6, June 2014.

[3] Rajeev Sobti and G.Geetha, "Cryptographic Hash Functions: A Review", International Journal of Computer Science Issues, Vol. 9, Issue 2, March 2012.

[4] I.Haitner, D.Harnik, and O.Reingold, "Efficient Pseudorandom Generators from Exponentially Hard OneWay Functions", in ICALP (2), 2006, pp.228-239.

[5] P.Gauravram, "Cryptographic Hash Functions: Cryptanalysis, design and Applications", Ph.D. Thesis, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia, 2003.

[6] Henry Gilbert, Helena Handschuh, "Security analysis of SHA-256 and Sisters",Lecture Notes in Computer Science 3006:175-19, 2003.

[7] P.Rogaway, and T.Shrimpton, "Cryptographic HashFunction Basics: Definitions, implications and separations for preimage resistance, second preimage resistance, and collision resistance",in Fast Software Encryption, 2004, pp.371-388.

[8] I.Damgard, "A Design Principle for Hash Functions", Advances in Cryptology — CRYPTO' 89 Proceedings, 1989, pp.416-427.

[9] X.Wang, X.Lai, D.Feng, H.Chen and X.Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD",In: Cramer R. (eds) Advances in Cryptology – EUROCRYPT 2005, pp.1-18.

[10] C.P.Schnorr, "An efficient Cryptographic Hash Functions", Advances in Cryptology — CRYPTO' 91 Proceedings, 1991.

Dr.T.Sivakumar is currently working as an Assistant Professor SG) in the Department of Information Technology, PSG College of Technology, Coimbatore-641004, India. His research interests include data & network security and cryptography.

Ms.Sowmya P is the final year students of B.Tech-I T, in the Department of Information Technology, PSG College of Technology, Coimbatore-641004, India

Ms.Utharaa S is the final year students of B.Tech-I T, in the Department of Information Technology, PSG College of Technology, Coimbatore-641004, India